

# QBUS6850

## Lecture 11

### Matrix Factorization

© *Discipline of Business Analytics*

BUSINESS SCHOOL

*Discipline of  
Business Analytics*



THE UNIVERSITY OF  
SYDNEY



## □ Topics covered

- Singular Value Decomposition (SVD)
- Matrix factorization (MF)
- Nonnegative matrix factorization (NMF)

## □ References

- Alpaydin, (2014), Chapter 6.7
- Lee and Seung (1999, 2001)
- Gillis (2014)

# Learning Objectives

- ❑ Understand how SVD works
- ❑ Understand the intuition of MF and NMF
- ❑ Understand how NMF work
- ❑ Understand the loss function of NMF
- ❑ Understand the two parameter estimation approaches of NMF
- ❑ Be able to apply NMF into real applications



# Matrix Factorization



# Matrix Factorization

$$\begin{bmatrix}
 -1.521 & -1.127 & -2.924 & 0.045 & 4.301 & 4.396 & -0.479 & -1.836 & 2.357 & 0.419 \\
 3.983 & 2.295 & 6.221 & -2.908 & -2.947 & 2.500 & -1.380 & 4.796 & -1.509 & -1.494 \\
 -0.706 & -1.712 & -1.057 & 0.018 & 3.717 & -0.813 & -0.890 & -2.865 & 1.866 & 1.834 \\
 -0.490 & -2.025 & -0.492 & 0.064 & 3.649 & -2.870 & -1.044 & -3.426 & 1.757 & 2.451 \\
 1.078 & 0.777 & 1.773 & -0.563 & -1.614 & -0.201 & -0.104 & 1.474 & -0.850 & -0.515 \\
 -0.263 & -1.331 & 0.140 & 0.631 & 0.734 & -5.463 & -0.212 & -2.487 & 0.194 & 1.931 \\
 0.630 & -0.338 & 0.669 & -1.318 & 2.813 & 3.502 & -1.308 & -0.091 & 1.519 & 0.331 \\
 0.101 & 0.920 & 0.358 & 0.656 & -3.244 & -1.935 & 1.040 & 1.294 & -1.703 & -0.887 \\
 -1.278 & -0.749 & -2.028 & 0.871 & 1.126 & -0.486 & 0.386 & -1.535 & 0.586 & 0.467 \\
 0.307 & -0.293 & 0.945 & 0.380 & -1.145 & -4.316 & 0.121 & -0.665 & -0.730 & 0.815 \\
 0.078 & 0.019 & 0.107 & -0.094 & 0.078 & 0.196 & -0.072 & 0.065 & 0.044 & -0.011 \\
 0.157 & 0.675 & -0.006 & -0.292 & -0.487 & 2.502 & 0.128 & 1.248 & -0.166 & -0.948 \\
 -1.812 & -0.375 & -3.043 & 1.246 & 0.579 & 0.905 & 0.940 & -1.019 & 0.371 & -0.279 \\
 0.400 & 1.145 & -0.005 & -0.988 & 0.222 & 6.340 & -0.192 & 2.305 & 0.314 & -1.736 \\
 -0.298 & -1.285 & -0.269 & 0.068 & 2.220 & -2.038 & -0.642 & -2.186 & 1.059 & 1.579 \\
 -1.305 & -1.757 & -2.066 & 0.460 & 3.717 & -0.490 & -0.511 & -3.080 & 1.893 & 1.647 \\
 0.769 & -1.241 & 1.676 & -0.478 & 1.637 & -4.047 & -1.141 & -1.959 & 0.669 & 2.074 \\
 -2.142 & -1.051 & -3.971 & 0.556 & 3.985 & 4.730 & 0.025 & -1.873 & 2.225 & 0.081 \\
 -2.245 & -1.167 & -3.568 & 1.589 & 1.611 & -0.821 & 0.808 & -2.468 & 0.845 & 0.643 \\
 0.622 & 0.689 & 0.917 & -0.405 & -1.069 & 0.908 & 0.010 & 1.289 & -0.525 & -0.666
 \end{bmatrix}
 =
 \begin{bmatrix}
 -1.407 & -1.793 & 0.328 \\
 -0.255 & 1.520 & -2.437 \\
 -1.348 & -0.492 & 1.083 \\
 -1.356 & -0.037 & 1.447 \\
 0.307 & 0.637 & -0.614 \\
 -0.109 & 0.770 & 1.416 \\
 -1.543 & -0.593 & -0.479 \\
 1.425 & 0.669 & -0.191 \\
 -0.004 & -0.559 & 0.745 \\
 0.477 & 0.981 & 0.634 \\
 -0.068 & -0.008 & -0.056 \\
 0.103 & -0.324 & -0.688 \\
 0.407 & -0.977 & 0.526 \\
 -0.380 & -0.950 & -1.484 \\
 -0.819 & 0.028 & 0.944 \\
 -1.130 & -0.802 & 1.218 \\
 -0.921 & 0.881 & 0.914 \\
 -1.043 & -2.086 & 0.411 \\
 0.153 & -0.954 & 1.236 \\
 0.216 & 0.202 & -0.603
 \end{bmatrix}
 \times
 \begin{bmatrix}
 -0.584 & 0.483 & -0.901 & 0.808 & -1.756 & -0.520 & 0.971 & 0.535 & -0.887 & -0.734 \\
 1.150 & 0.076 & 2.131 & -0.515 & -0.865 & -2.507 & -0.462 & 0.264 & -0.551 & 0.529 \\
 -0.856 & -0.944 & -1.129 & 0.788 & 0.854 & -2.535 & 0.176 & -1.859 & 0.368 & 1.019
 \end{bmatrix}$$



# Why Matrix?

-1.521	-1.127	-2.924	0.045	4.301	4.396	-0.479	-1.836	2.357	0.419
3.983	2.295	6.221	-2.908	-2.947	2.500	-1.380	4.796	-1.509	-1.494
-0.706	-1.712	-1.057	0.018	3.717	-0.813	-0.890	-2.865	1.866	1.834
-0.490	-2.025	-0.492	0.064	3.649	-2.870	-1.044	-3.426	1.757	2.451
1.078	0.777	1.773	-0.563	-1.614	-0.201	-0.104	1.474	-0.850	-0.515
-0.263	-1.331	0.140	0.631	0.734	-5.463	-0.212	-2.487	0.194	1.931
0.630	-0.338	0.669	-1.318	2.813	3.502	-1.308	-0.091	1.519	0.331
0.101	0.920	0.358	0.656	-3.244	-1.935	1.040	1.294	-1.703	-0.887
-1.278	-0.749	-2.028	0.871	1.126	-0.486	0.386	-1.535	0.586	0.467
0.307	-0.293	0.945	0.380	-1.145	-4.316	0.121	-0.665	-0.730	0.815
0.078	0.019	0.107	-0.094	0.078	0.196	-0.072	0.065	0.044	-0.011
0.157	0.675	-0.006	-0.292	-0.487	2.502	0.128	1.248	-0.166	-0.948
-1.812	-0.375	-3.043	1.246	0.579	0.905	0.940	-1.019	0.371	-0.279
0.400	1.145	-0.005	-0.988	0.222	6.340	-0.192	2.305	0.314	-1.736
-0.298	-1.285	-0.269	0.068	2.220	-2.038	-0.642	-2.186	1.059	1.579
-1.305	-1.757	-2.066	0.460	3.717	-0.490	-0.511	-3.080	1.893	1.647
0.769	-1.241	1.676	-0.478	1.637	-4.047	-1.141	-1.959	0.669	2.074
-2.142	-1.051	-3.971	0.556	3.985	4.730	0.025	-1.873	2.225	0.081
-2.245	-1.167	-3.568	1.589	1.611	-0.821	0.808	-2.468	0.845	0.643
0.622	0.689	0.917	-0.405	-1.069	0.908	0.010	1.289	-0.525	-0.666

- Common digital representations
- Solid math support
- Facilitate computer implementation
- ...



$$X = W \times H$$

-1.521	-1.127	-2.924	0.045	4.301	4.396	-0.479	-1.836	2.357	0.419
3.983	2.295	6.221	-2.908	-2.947	2.500	-1.380	4.796	-1.509	-1.494
-0.706	-1.712	-1.057	0.018	3.717	-0.813	-0.890	-2.865	1.866	1.834
-0.490	-2.025	-0.492	0.064	3.649	-2.870	-1.044	-3.426	1.757	2.451
1.078	0.777	1.773	-0.563	-1.614	-0.201	-0.104	1.474	-0.850	-0.515
-0.263	-1.331	0.140	0.631	0.734	-5.463	-0.212	-2.487	0.194	1.931
0.630	-0.338	0.669	-1.318	2.813	3.502	-1.308	-0.091	1.519	0.331
0.101	0.920	0.358	0.656	-3.244	-1.935	1.040	1.294	-1.703	-0.887
-1.278	-0.749	-2.028	0.871	1.126	-0.486	0.386	-1.535	0.586	0.467
0.307	-0.293	0.945	0.380	-1.145	-4.316	0.121	-0.665	-0.730	0.815
0.078	0.019	0.107	-0.094	0.078	0.196	-0.072	0.065	0.044	-0.011
0.157	0.675	-0.006	-0.292	-0.487	2.502	0.128	1.248	-0.166	-0.948
-1.812	-0.375	-3.043	1.246	0.579	0.905	0.940	-1.019	0.371	-0.279
0.400	1.145	-0.005	-0.988	0.222	6.340	-0.192	2.305	0.314	-1.736
-0.298	-1.285	-0.269	0.068	2.220	-2.038	-0.642	-2.186	1.059	1.579
-1.305	-1.757	-2.066	0.460	3.717	-0.490	-0.511	-3.080	1.893	1.647
0.769	-1.241	1.676	-0.478	1.637	-4.047	-1.141	-1.959	0.669	2.074
-2.142	-1.051	-3.971	0.556	3.985	4.730	0.025	-1.873	2.225	0.081
-2.245	-1.167	-3.568	1.589	1.611	-0.821	0.808	-2.468	0.845	0.643
0.622	0.689	0.917	-0.405	-1.069	0.908	0.010	1.289	-0.525	-0.666

-1.407	-1.793	0.328
-0.255	1.520	-2.437
-1.348	-0.492	1.083
-1.356	-0.037	1.447
0.307	0.637	-0.614
-0.109	0.770	1.416
-1.543	-0.593	-0.479
1.425	0.669	-0.191
-0.004	-0.559	0.745
0.477	0.981	0.634
-0.068	-0.008	-0.056
0.103	-0.324	-0.688
0.407	-0.977	0.526
-0.380	-0.950	-1.484
-0.819	0.028	0.944
-1.130	-0.802	1.218
-0.921	0.881	0.914
-1.043	-2.086	0.411
0.153	-0.954	1.236
0.216	0.202	-0.603

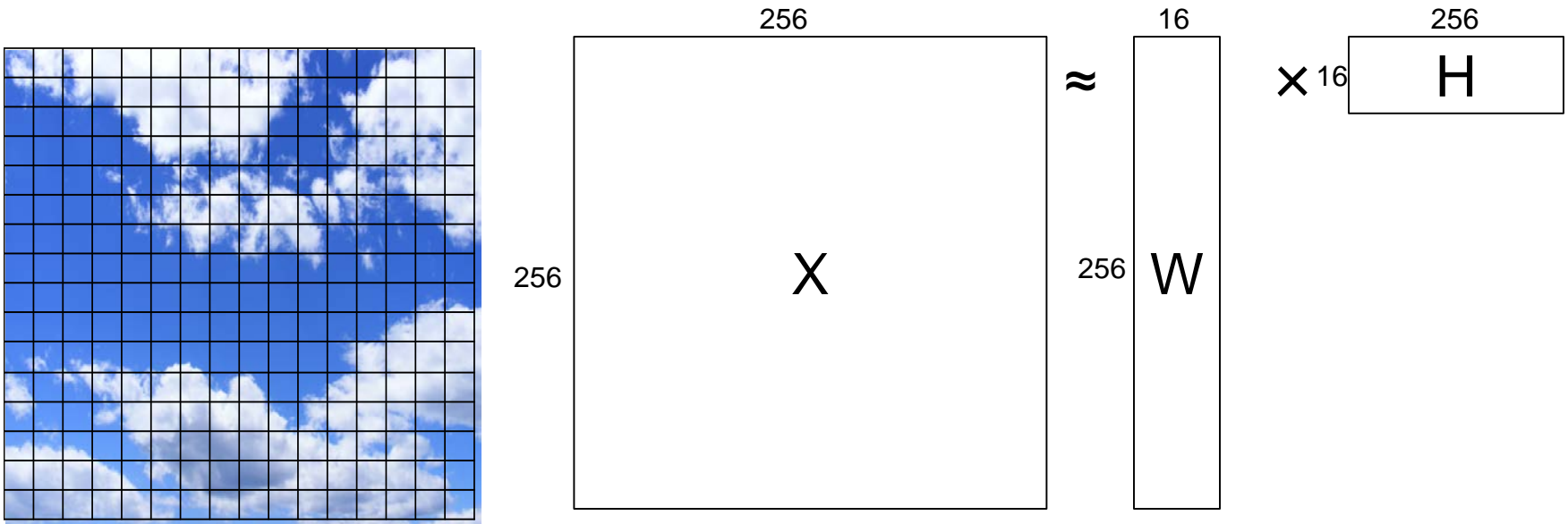
-0.584	0.483	-0.901	0.808	-1.756	-0.520	0.971	0.535	-0.887	-0.734
1.150	0.076	2.131	-0.515	-0.865	-2.507	-0.462	0.264	-0.551	0.529
-0.856	-0.944	-1.129	0.788	0.854	-2.535	0.176	-1.859	0.368	1.019

$$(WH)_{ij} = \sum_k W_{ik} H_{kj}$$



# Why Matrix Factorization?

## Compression



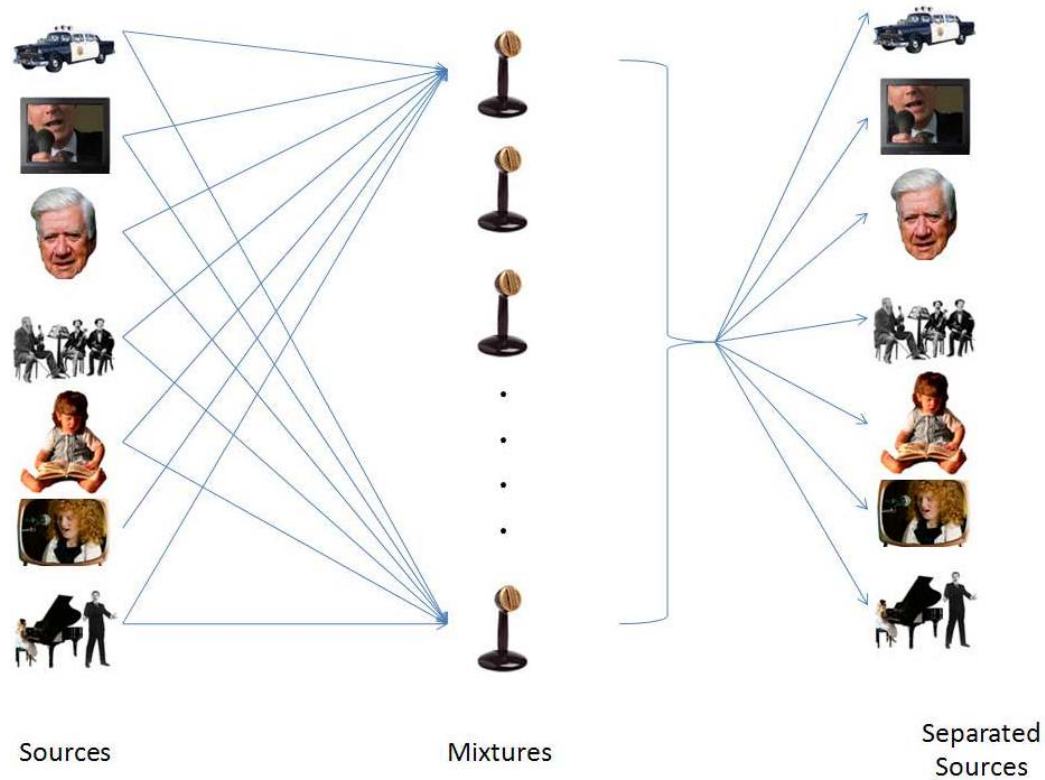
$$256 \times 256 = 65536$$

$$256 \times 16 + 16 \times 256 = 8192$$





## Source separation



<https://onionesquereality.wordpress.com/2010/01/30/blind-source-separation-in-magnetic-resonance-images/>



# Singular Value Decomposition



# SVD

$$\mathbf{X} \in \mathbb{R}^{n \times m}, \mathbf{W} \in \mathbb{R}^{n \times r}, \mathbf{H} \in \mathbb{R}^{r \times m}, r \ll \min(n, m)$$

$$\underset{\mathbf{W}, \mathbf{H}}{\text{minimize}} \quad \|\mathbf{X} - \mathbf{WH}\|_F^2 = \sum_{ij} [X_{ij} - (\mathbf{WH})_{ij}]^2$$

□ **By Singular Value Decomposition (SVD)**

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$


$$\mathbf{W} = \mathbf{U}$$

$$\mathbf{H} = \mathbf{\Sigma}\mathbf{V}^T$$

# SVD

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$



- $\mathbf{U}$  is an  $n \times n$  unitary matrix.  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ . Columns are orthogonal unit vectors.
- $\mathbf{\Sigma}$  is a diagonal  $n \times m$  matrix with non-negative real numbers on the diagonal
- $\mathbf{V}$  is an  $m \times m$  unitary matrix.  $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ . Columns are orthogonal unit vectors.
- The diagonal entries  $\sigma_i$  of  $\mathbf{\Sigma}$  known as the singular values of  $\mathbf{X}$ . 
- A common convention is to list the singular values in descending order. In this case, the diagonal matrix,  $\mathbf{\Sigma}$  is uniquely determined by  $\mathbf{X}$ .
- Since  $\mathbf{U}$  and  $\mathbf{V}$  are unitary, the columns of each of them form a set of orthonormal vectors, which can be regarded as basis vectors.

$$\mathbf{X}^T \mathbf{X} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = (\mathbf{V}\mathbf{\Sigma}^T \mathbf{U}^T) \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T (\mathbf{U}^T \mathbf{U}) \mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T \mathbf{\Sigma}\mathbf{V}^T$$





# Example

$$\mathbf{X} = [\mathbf{u}_1 \quad \mathbf{u}_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix}$$


$$\mathbf{X} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}$$

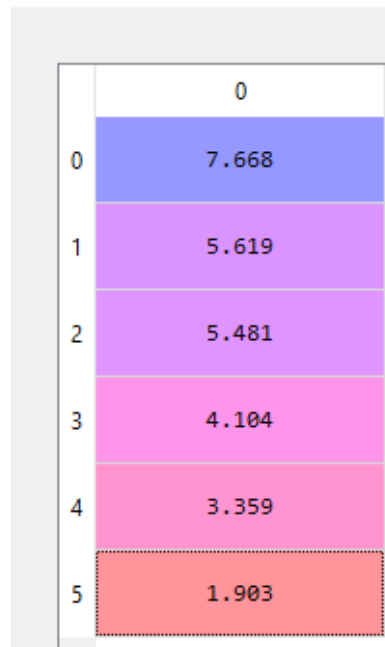
$$\mathbf{X} = \begin{bmatrix} -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

# Python Example

```
XX= np.random.randn(20, 6)  
U, s, V = np.linalg.svd(XX, full_matrices=True)  
U.shape, V.shape, s.shape
```

```
((20, 20), (6, 6), (6,))
```

 s - NumPy array



The singular values sorted in descending order.



# Nonnegative Matrix Factorization

# NMF

- Non-Negative Matrix Factorization (NMF) is a recent technique for linear dimensionality reduction and data analysis that yields a parts based, sparse non-negative representation for non-negative input data.
- Essentially, NMF is an unsupervised learning algorithm performs data dimensionality reduction and clustering simultaneously.
- Having a non-negative data matrix containing a set of observations with a fixed number of features, NMF algorithm will produce a factorization of the data matrix and reveal the interesting latent factors underlying the interactions between observations and their features.



# Nonnegative Data

Lots of data in real life are nonnegative:

- Pixel intensities;
- Amplitude spectra;
- Occurrence counts;
- Food or energy consumption;
- User scores;
- Stock market indices;
- ...

For the sake of interpretability of the results, optimal processing of nonnegative data may call for processing under nonnegativity constraints.



# NMF History

NMF is more than 30-year old!

- ❑ previous variants referred to as:
  - nonnegative rank factorization (Jeter and Pye, 1981; Chen, 1984);
  - positive matrix factorisation (Paatero and Tapper, 1994);
- ❑ popularized by Lee and Seung (1999) for “learning the parts of objects”.

Since then, widely used in various research areas for diverse applications.

---

## Algorithms for Non-negative Matrix Factorization

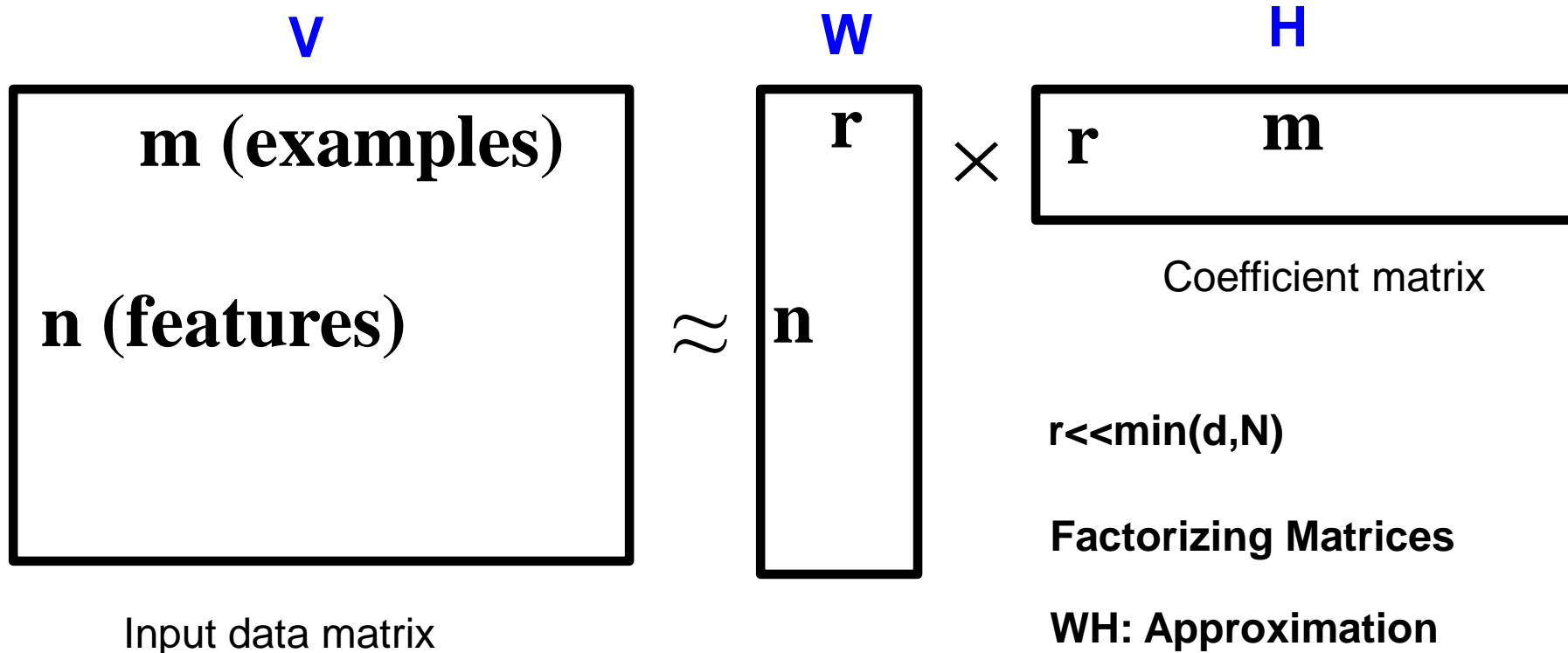
---

**Daniel D. Lee\***

\*Bell Laboratories  
Lucent Technologies  
Murray Hill, NJ 07974

**H. Sebastian Seung\*<sup>†</sup>**

<sup>†</sup>Dept. of Brain and Cog. Sci.  
Massachusetts Institute of Technology  
Cambridge, MA 02138



The columns of **W** and the rows of **H** represent the latent factor bundles that are believed to be responsible for the observed data in **V**.



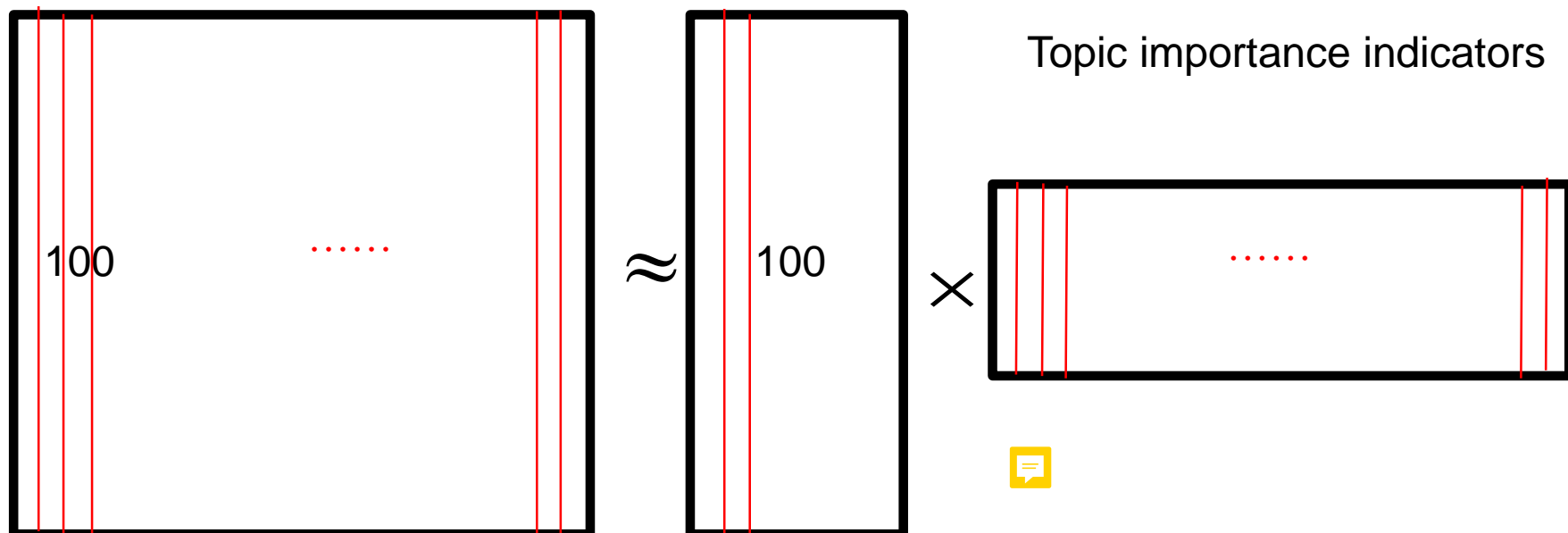
# Why NMF?

## Topics/semantics learning

Documents

10 Topics

Topic importance indicators



# Clustering

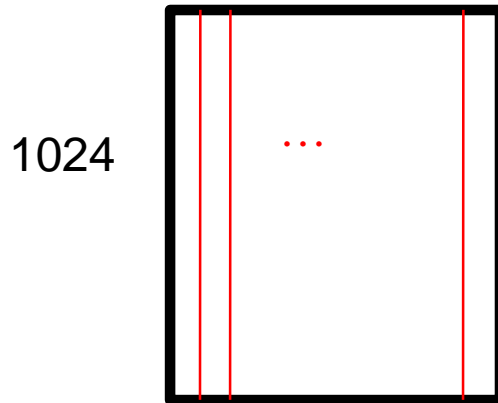




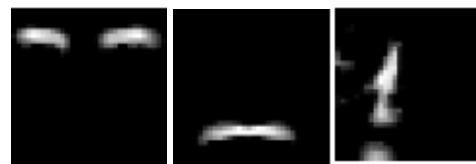
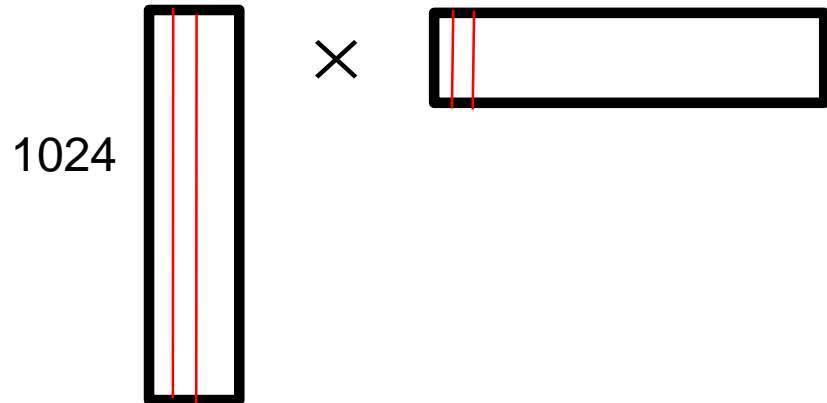
# Why NMF?

Learning the part-based representations (sparse), feature learning

256 vectorized images



16 basis images.  
Facial features.



- Data compression
- Text mining, etc

# NMF Applications

NMF is a non-supervised data decomposition technique, akin to latent variable analysis, that can be used for:

- Topics recovery: like Probabilistic Latent Semantic Analysis (PLSA);
- Feature learning: like Principal Component Analysis (PCA);
- Clustering: like K-means;
- Temporal segmentation: like Hidden Markov Models (HMM);
- Filtering and source separation: as with Independent Component Analysis (ICA);
- Coding as with vector quantization;
- Stock market.

# NMF Model

- Non-negative matrix factorization (NMF) has previously been shown to be a useful decomposition method for multivariate data. In NMF, given a non-negative matrix  $\mathbf{V}$ , find non-negative matrix factors  $\mathbf{W}$  and  $\mathbf{H}$  such that:
- $\mathbf{V} = n \times m$ , matrix of the original data,
- $\mathbf{W} = n \times r$ , matrix contains “basis vectors”,
- $\mathbf{H} = r \times m$ , coefficient matrix, and we have  $r < \min(n, m)$ .

$$\mathbf{V} \approx \mathbf{WH}$$



# NMF Model

**V** : the  $n \times m$  data matrix:

- $n$  features (rows),
- $m$  observations/examples vectors (columns),
- $\mathbf{v}_j = \mathbf{V}(:, j) = [v_{1j}, v_{2j}, v_{3j}, \dots, v_{nj}]^T$ ,  $n \times 1$  column vector, the  $j_{th}$  observation vector,  $1 \leq j \leq m$

**W** : the  $n \times r$  dictionary matrix:

- $w_{ik}$  is one of its coefficients,
- $\mathbf{w}_k = \mathbf{W}(:, k) = [w_{1k}, w_{2k}, w_{3k}, \dots, w_{nk}]^T$ ,  $n \times 1$  column vector, a dictionary/basis vector,  $1 \leq k \leq r$

**H** : the  $r \times m$  activation/expansion matrix:

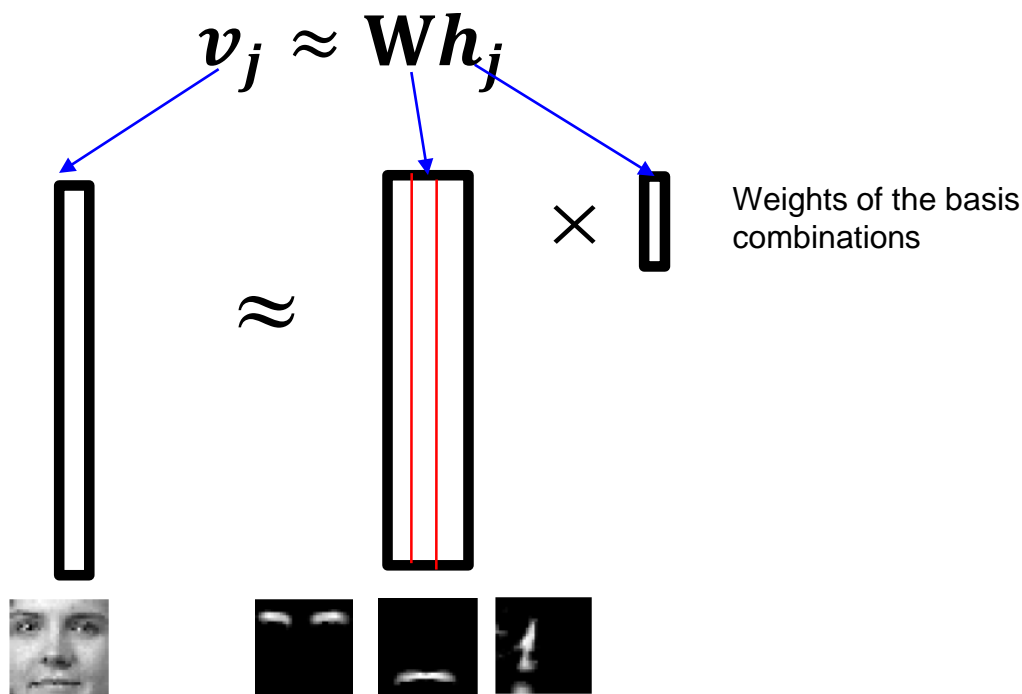
- $\mathbf{h}_j = \mathbf{H}(:, j) = [h_{1j}, h_{2j}, h_{3j}, \dots, h_{rj}]^T$ , the  $r \times 1$  column vector of activation coefficients/each column of **H** gives the coordinates of observation  $\mathbf{v}_j$ ,  $1 \leq j \leq m$



# NMF Model

$\mathbf{V}$  is decomposed into a tall & skinny matrix  $\mathbf{W}$  and a short & wide matrix  $\mathbf{H}$

Each column of  $\mathbf{V}$ ,  $v_j$ , can be calculate as:





# Python Example

V

5	3	0	1
4	0	0	1
1	1	0	5
1	0	0	4
0	1	5	4

≈

W

0.000	2.375
0.000	1.583
1.357	0.593
1.063	0.443
1.976	0.000

×

H

0.000	0.329	1.437	2.641
2.213	0.839	0.000	0.613



# Python Example

**W**

0.000	2.375
0.000	1.583
1.357	0.593
1.063	0.443
1.976	0.000

**×**

**H**

0.000	0.329	1.437	2.641
2.213	0.839	0.000	0.613

**=**

5.256	1.993	0.000	1.455
3.504	1.329	0.000	0.970
1.313	0.944	1.950	3.946
0.981	0.722	1.528	3.079
0.000	0.650	2.840	5.219

This matrix is an approximation  
of the matrix **V**

# Example

- Let each column of the data matrix  $\mathbf{V}$  ( $n \times m$ ) be a vectorized gray-level image of a face, with the  $(i, j)_{th}$  entry of matrix  $\mathbf{V}$  being the intensity of the  $i_{th}$  pixel in the  $j_{th}$  face.
- NMF generates two matrices ( $\mathbf{W}$ ,  $\mathbf{H}$ ) so that each image  $\mathbf{V}(:, j)$  is approximated using a linear combination of the columns of  $\mathbf{W}$  (facial features)
- Since  $\mathbf{W}$  is nonnegative, the columns of  $\mathbf{W}$  can be interpreted as images (that is, vectors of pixel intensities) which we refer to as the basis images, e.g. nose, eye brow, lips, moustaches, etc. The matrix is sparse.
- As the weights in the linear combinations are nonnegative ( $\mathbf{H} \geq 0$ ), these basis images can only be summed up to reconstruct each original image.
- The columns of  $\mathbf{H}$  indicate which feature (and its weights) is present in which image. For example, image 1 has nose, eye brow, lips, and no moustaches.



# Example

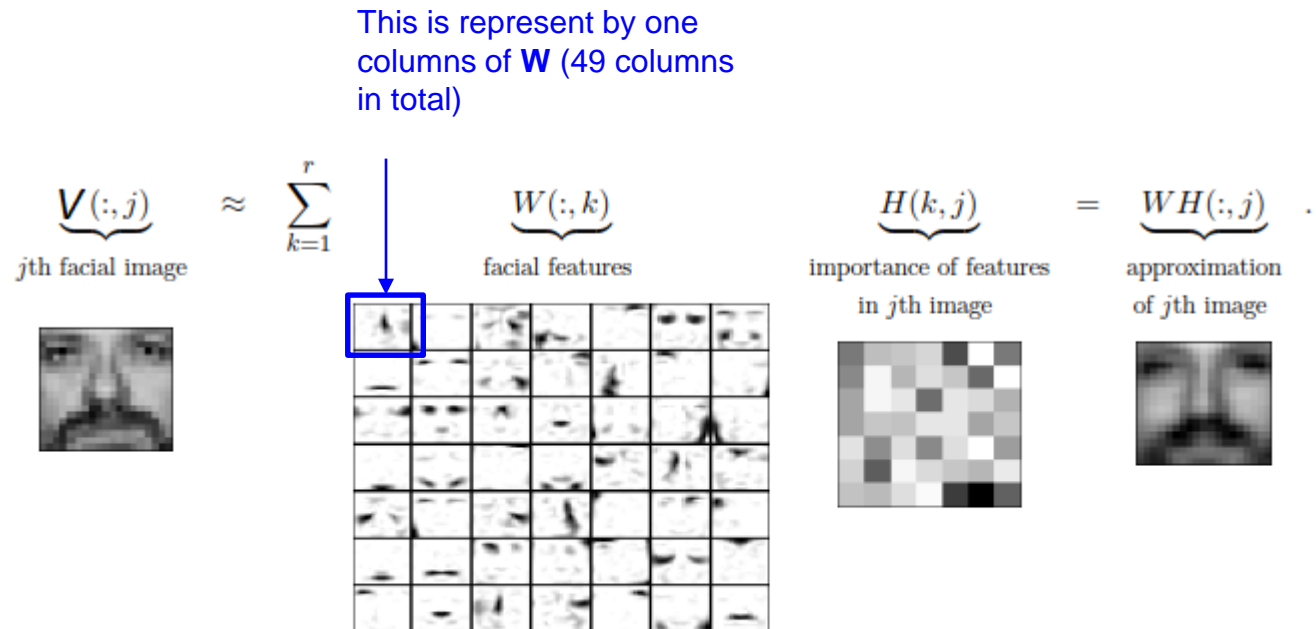
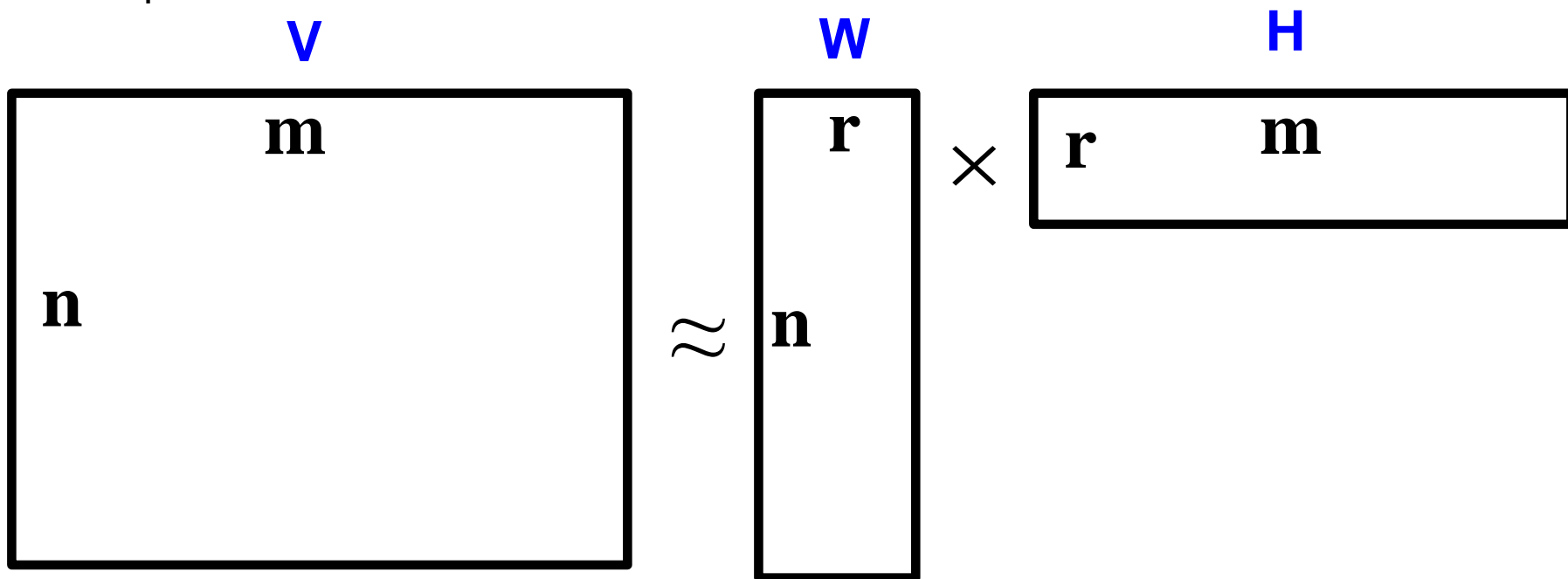


Figure 1: Decomposition of the CBCL face database, MIT Center For Biological and Computation Learning (2429 gray-level 19-by-19 pixels images) using  $r = 49$  as in [77].



# NMF Model

NMF is an unsupervised linear algorithm, and NMF produces part-based representations of the data.

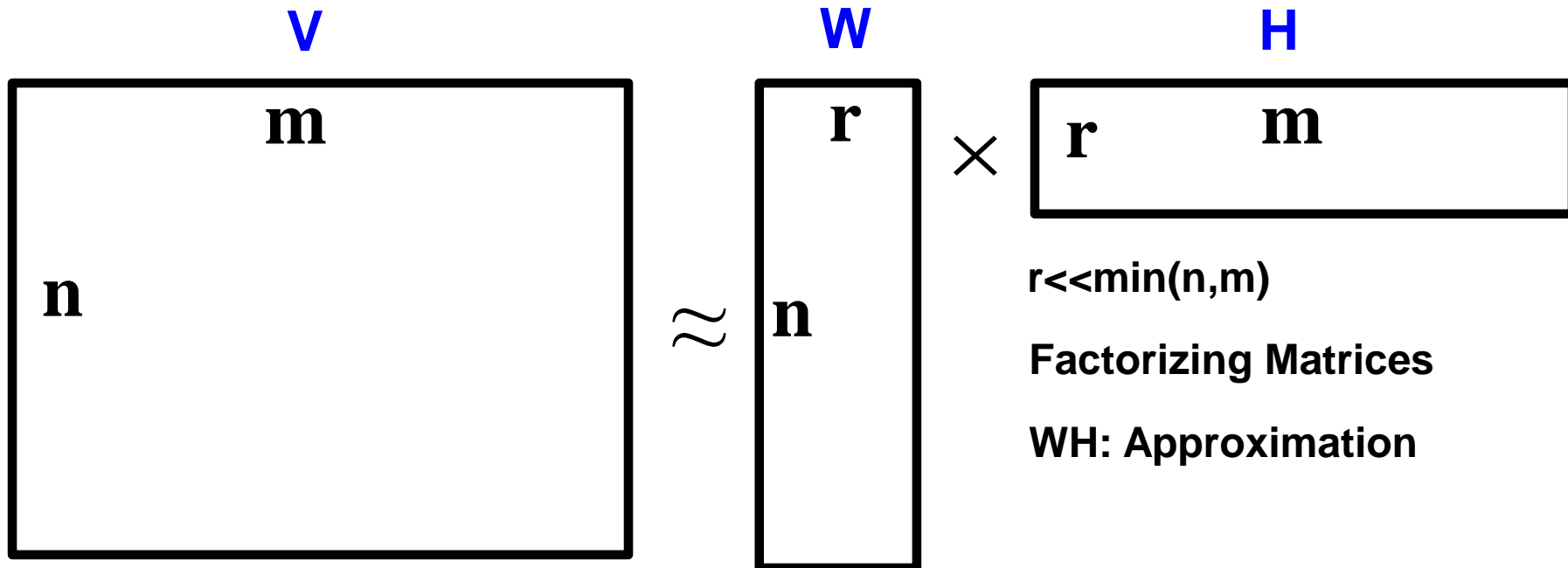


$$V \approx WH$$

$$W = [W_{ik}], \text{ s.t. } W_{ik} \geq 0,$$

$$H = [H_{kj}], \text{ s.t. } H_{kj} \geq 0,$$

# NMF



**Loss function:** the approximation error between the input matrix and its approximation.  $D()$  is a separable matrix divergence.

$$L(\mathbf{V}, \mathbf{WH}) = D(\mathbf{V} \mid \mathbf{WH})$$





# NMF Loss Function

We consider two optimization problems, using loss functions:

## (a) Euclidean distance square

Minimize  $L(V, WH) = \|V - WH\|^2 = \sum_{ij} (V_{ij} - (WH)_{ij})^2$ ,

w.r.t  $W$  and  $H$ ,

subject to the constraints  $W \geq 0, H \geq 0$ .

## (b) Divergence

b. Minimize  $L(V, WH) = \sum_{ij} \left( V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right)^2$ ,

w.r.t  $W$  and  $H$ ,

subject to the constraints  $W \geq 0, H \geq 0$ .

# NMF Estimation- Approach1

$$\mathbf{V} \approx \mathbf{W}\mathbf{H}$$

$$\mathbf{W} = [W_{ik}], \text{ s.t. } W_{ik} \geq 0,$$

$$\mathbf{H} = [H_{kj}], \text{ s.t. } H_{kj} \geq 0,$$

$L(\mathbf{V}, \mathbf{W}\mathbf{H})$  is the given cost function, then update the parameters according to the gradient descent algorithm:

$$\mathbf{W}_{ik} := \mathbf{W}_{ik} - \mu_{ik} \frac{\partial L}{\partial \mathbf{W}_{ik}},$$

$$\mathbf{H}_{kj} := \mathbf{H}_{kj} - \nu_{kj} \frac{\partial L}{\partial \mathbf{H}_{kj}}$$



# Regularization

The above algorithm is a very basic algorithm for factorizing a matrix. There are a lot of methods to make things look more complicated. A common extension to this basic algorithm is to introduce regularization to avoid overfitting.

When L1 regularization (Lasso) is added to NMF with the mean squared loss function, the resulting problem may be called non-negative sparse coding (Hoyer, 2002) due to the similarity to the sparse coding problem, although it may also still be referred to as NMF.

Hoyer, Patrik O. (2002). Non-negative sparse coding. Proc. IEEE Workshop on Neural Networks for Signal Processing.



# NMF Estimation- Approach 2

$$\min_{\mathbf{W}, \mathbf{H} \geq 0} L(\mathbf{V}, \mathbf{WH}) = \|\mathbf{V} - \mathbf{WH}\|^2$$

NMF can employ **multiplicative update rules** on and to get the optimal solution which minimizes the objective, the update rules are (Lee and Seung, 2001):

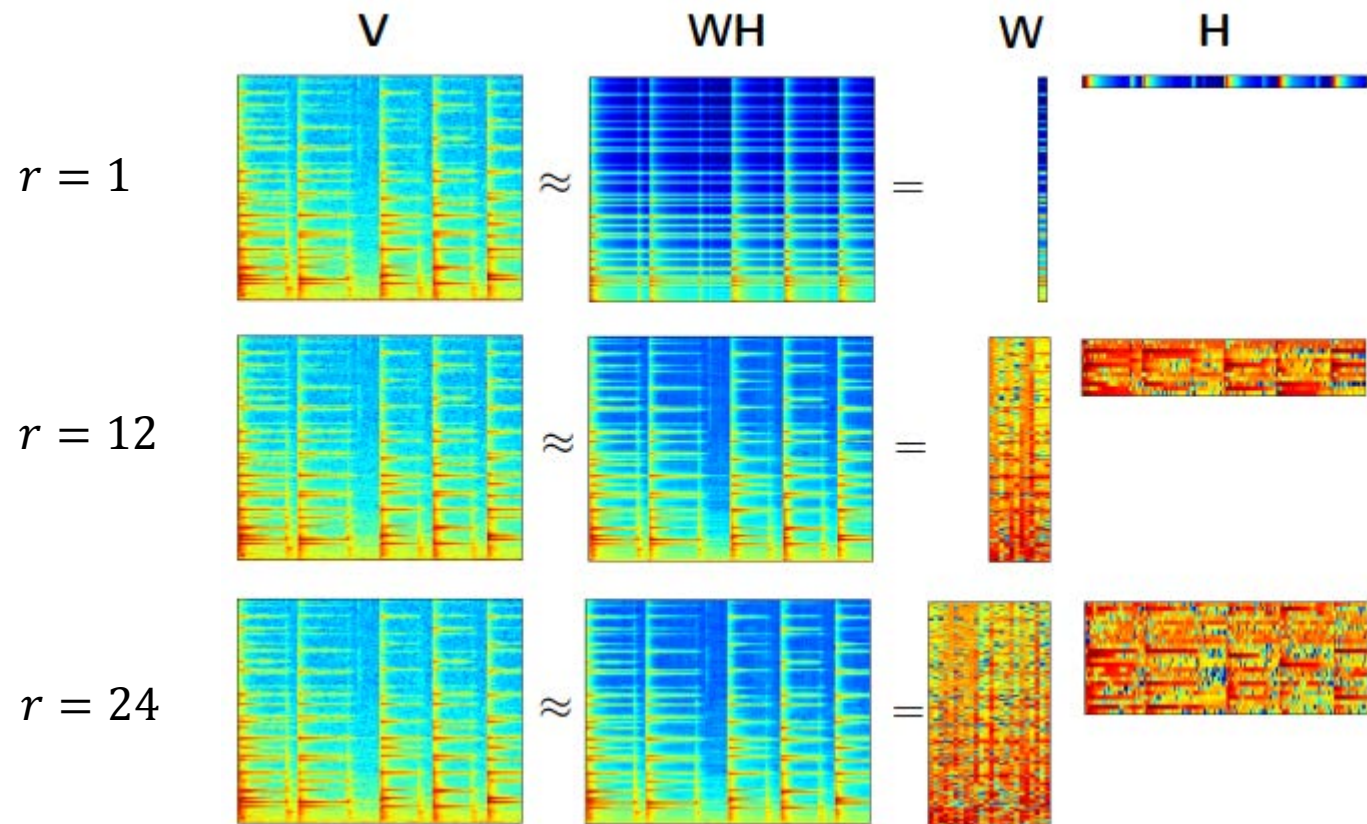
$$\mathbf{H}_{kj} := \mathbf{H}_{kj} \frac{(\mathbf{W}^T \mathbf{V})_{kj}}{(\mathbf{W}^T \mathbf{WH})_{kj}}$$

$$\mathbf{W}_{ik} := \mathbf{W}_{ik} \frac{(\mathbf{VH}^T)_{ik}}{(\mathbf{WHH}^T)_{ik}}$$

**NOTE:** The nonnegativity of  $\mathbf{W}$  and  $\mathbf{H}$  is guaranteed by construction.

# How to choose $r$ ?

- ❑ The choice of  $r$  results in a compromise between
  - Data fitting
  - A greater  $r$  leads to a better data approximation
  - Model complexity
  - A smaller  $r$  leads to a less complex model (easier to estimate, less parameters to transmit, etc ...)
  
- ❑ A right model order choice is important and it depends on the data  $V$  and on the application.



# How to choose $r$ ?

The following strategies are usually used to set up an appropriate model order:

- **Model order  $r$  is fixed** during the NMF decomposition, and it was
  - i. chosen by intuition,
  - ii. chosen based on some prior knowledge (e.g., known number of clusters for clustering),
  - iii. Estimation using the SVD: look at the decay of the singular values of the input data matrix
  - iv. or trained on some development data within a particular application.
- **Model order  $r$  is estimated automatically** within the NMF Decomposition (Tan and Févotte, 2013; Schmidt and Morup, 2010).

# W and H initialization

A good initialization of parameters (**W** and **H**) is important for any local optimization approach (including multiplicative update rules) due to the existence of many local minima.

## Random initializations:

- Initialize (nonnegative) parameters randomly several times;
- Keep the solution with the lowest final loss.

## Structural data-driven initializations:

- Initialize **W** by clustering of data points **V** (Kim and Choi, 2007);
- Initialize **W** by singular value decomposition (SVD) (Boutsidis and Gallopoulos, 2008);



# Stopping Criteria

For any iterative optimization strategy (including MU rules) the total number of iterations is important and results in a tradeoff between:

- The computational load from one side,
- The data fitting (approximation error) and model quality from the other side.

**Stopping criteria** (Albright et al., 2006):

- After a fixed number of iterations;
- Once the approximation error (the loss) is below a pre-defined threshold;
- Once the approximation error relative decrease is below a pre-defined threshold



# Improve NMF Performance

## Problems:

- NMF is **not unique**.
- Hence NMF is not guaranteed to extract latent components as desired within a particular application.

## Possible solution:

Given the application, impose some knowledge-based constraints on  $\mathbf{W}$ , on  $\mathbf{H}$ , or on both  $\mathbf{W}$  and  $\mathbf{H}$ .

- Adding constraints usually makes the decomposition “more unique”.
- Appropriate constraints may lead to more suitable latent components.



# NMF Applications



# Feature Extraction

Dataset: 'FERET face'. Size: 1024\*2409. 2409 'face images' with size 32\*32 (reshape: 1024 dimensional column vector)

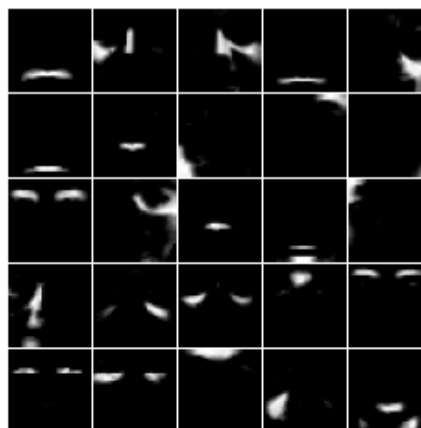
NMF and Projective NMF  
(for your interest) for  
Feature Extraction



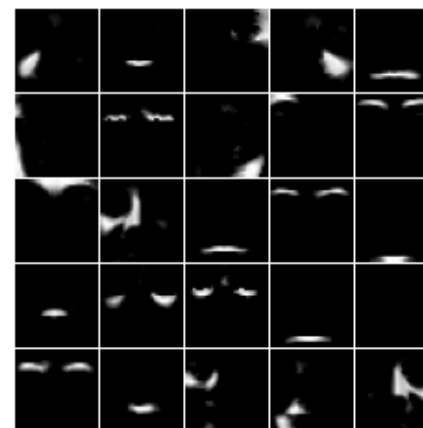
(a) original dataset



(b) Linear EU-NMF



(c) EU-PNMF



(d)  $\alpha$ -PNMF ( $\alpha = 2$ )



# Topic Modelling

- Let each column of the nonnegative data matrix  $\mathbf{V}$  ( $n \times m$ ) correspond to a document and each row to a word.
- The  $(i, j)_{th}$  entry of the matrix  $\mathbf{V}$  could for example be equal the number of times the  $i_{th}$  word appears in the  $j_{th}$  document in which case each column of  $\mathbf{V}$  is the vector of word counts of a document
- In practice, more sophisticated constructions are used, e.g., the term frequency – inverse document frequency (tf-idf).
- $\mathbf{V}$  is very sparse matrix, as most documents only use a small subset of the dictionary/words/features listed

$$\underbrace{V(:, j)}_{j\text{th document}} \approx \sum_{k=1}^r \underbrace{W(:, k)}_{k\text{th topic}} \underbrace{H(k, j)}_{\substack{\text{importance of } k\text{th topic} \\ \text{in } j\text{th document}}}, \quad \text{with } W \geq 0 \text{ and } H \geq 0.$$



# Topic Modelling

$$\underbrace{X(:, j)}_{j\text{th document}} \approx \sum_{k=1}^r \underbrace{W(:, k)}_{k\text{th topic}} \underbrace{H(k, j)}_{\substack{\text{importance of } k\text{th topic} \\ \text{in } j\text{th document}}}, \quad \text{with } W \geq 0 \text{ and } H \geq 0.$$

- **W** is nonnegative, each column of **W** can be interpreted as a document, that is, as a bag of words.
- The weights in the linear combinations are nonnegative (**H** ≥ 0), one can only take the union of the sets of words defined by the columns of **W** to reconstruct all the original documents.
- Because the number of documents (m) in the data set is much larger than the number of basis elements (the number of columns of **W**: r). The latter should be set of words found simultaneously in several documents. Hence the basis elements can be interpreted as topics, that is, set of words found simultaneously in different documents
- The weights in the linear combinations (matrix **H**) assign the documents to the different topics: identify which document discusses which topic. For example, document 1 has high weight for topic 5 and low weights for all other topics



# Example

**V**

	Doc 1	Doc 2	Doc 3	Doc 4
Word 1	36	3	45	54
Word 2	4	34	23	31
Word 3	9	65	11	0
Word 4	17	3	3	0
Word 5	0	14	7	4

$\approx$

**W**

**x**

**H**

	Topic 1	Topic 2
Word 1	0.0211	0.6341
Word 2	0.2689	0.2425
Word 3	0.5652	0.0000
Word 4	0.0281	0.0883
Word 5	0.1167	0.0351

	Doc 1	Doc 2	Doc 3	Doc 4
Topic 1	10.98	118.17	21.25	0.00
Topic 2	55.02	0.83	67.75	89.00



# Stock Market

NMF can be applied to stock trading volumes/prices

Matrix **V**: each column represents the total volume/closing prices for a stock; each rows represents each day

Matrix **W**: each column represents the latent features/basis

Matrix **H**: each column represents weights of the linear combinations of basis